

- [88] M. T. Aga and T. M. Austin, “Smokestack: Thwarting DOP Attacks with Runtime Stack Layout Randomization,” in *IEEE/ACM International Symposium on Code Generation and Optimization, CGO*, pp. 26–36, 2019.
- [89] S. Lee, H. Kang, J. Jang, and B. B. Kang, “SaVioR: Thwarting Stack-based Memory Safety Violations by Randomizing Stack Layout,” *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 4, pp. 2559–2575, 2022.
- [90] Y. Younan, D. Pozza, F. Piessens, and W. Joosen, “Extended Protection against Stack Smashing Attacks without Performance Loss,” in *22nd Annual Computer Security Applications Conference (ACSAC 2006), 11-15 December 2006, Miami Beach, Florida, USA*, pp. 429–438, IEEE Computer Society, 2006.
- [91] Y. Xu, Y. Solihin, and X. Shen, “MERR: Improving Security of Persistent Memory Objects via Efficient Memory Exposure Reduction and Randomization,” in *ASPLOS ’20: Architectural Support for Programming Languages and Operating Systems*, pp. 987–1000, 2020.
- [92] G. S. Kc, A. D. Keromytis, and V. Prevelakis, “Countering Code-injection Attacks With Instruction-set Randomization,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS*, pp. 272–280, 2003.
- [93] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, and D. D. Zovi, “Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS*, pp. 281–289, 2003.
- [94] M. Chew and D. Song, “Mitigating Buffer Overflows by Operating System Randomization,” Tech. Rep. CS-02-197, Carnegie Mellon University, 2002.
- [95] D. Couroussé, T. Barry, B. Robisson, P. Jaillon, O. Potin, and J. Lanet, “Runtime Code Polymorphism as a Protection Against Side Channel Attacks,” in *Proceedings of Information Security Theory and Practice - 10th IFIP WG 11.2 International Conference, WISTP*, vol. 9895, pp. 136–152, 2016.
- [96] S. Cao, N. He, Y. Guo, and H. Wang, “WASMixer: Binary Obfuscation for WebAssembly,” *CoRR*, vol. abs/2308.03123, 2023.
- [97] C. Collberg, C. Thomborson, and D. Low, “A taxonomy of obfuscating transformations,” tech. rep., Department of Computer Science, The University of Auckland, New Zealand, 1997.
- [98] M. Jacob, M. H. Jakubowski, P. Naldurg, C. W. Saw, and R. Venkatesan, “The Superdiversifier: Peephole Individualization for Software Protection,”

- in *Proceedings of Advances in Information and Computer Security, Third International Workshop on Security, IWSEC 2008*, vol. 5312, pp. 100–120, 2008.
- [99] M. Henry, “Superoptimizer: A Look at the Smallest Program,” *ACM SIGARCH Computer Architecture News*, vol. 15, pp. 122–126, Nov 1987.
 - [100] V. Le, M. Afshari, and Z. Su, “Compiler Validation via Equivalence Modulo Inputs,” in *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI*, pp. 216–226, 2014.
 - [101] B. R. Churchill, O. Padon, R. Sharma, and A. Aiken, “Semantic Program Alignment for Equivalence Checking,” in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI*, pp. 1027–1040, 2019.
 - [102] V. Le, C. Sun, and Z. Su, “Finding Deep Compiler Bugs via Guided Stochastic Program Mutation,” in *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, p. 386–399, 2015.
 - [103] E. Schulte, Z. P. Fry, E. Fast, W. Weimer, and S. Forrest, “Software mutational robustness,” vol. 15, p. 281–312, sep 2014.
 - [104] B. Baudry, S. Allier, and M. Monperrus, “Tailored source code transformations to synthesize computationally diverse program variants,” *ISSTA 2014*, p. 149–159, 2014.
 - [105] M. Zalewski, “American Fuzzy Lop,” 2017.
 - [106] K. Zhang, D. Wang, J. Xia, W. Y. Wang, and L. Li, “ALGO: Synthesizing Algorithmic Programs with Generated Oracle Verifiers,” *CoRR*, vol. abs/2305.14591, 2023.
 - [107] L. de Moura and N. Bjørner, “Z3: An Efficient SMT Solver,” in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, 2008.
 - [108] A. Abate, C. David, P. Kesseli, D. Kroening, and E. Polgreen, “Counterexample Guided Inductive Synthesis Modulo Theories,” in *Proceedings of Computer Aided Verification - 30th International Conference, CAV*, vol. 10981, pp. 270–288, 2018.
 - [109] P. M. Phothilimthana, A. Thakur, R. Bodík, and D. Dhurjati, “Scaling up Superoptimization,” in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS*, pp. 297–310, 2016.